Physically Unclonable Function (PUF)

Based on the tutorial by Dr. Kent Chuang at COSIC Course 2019

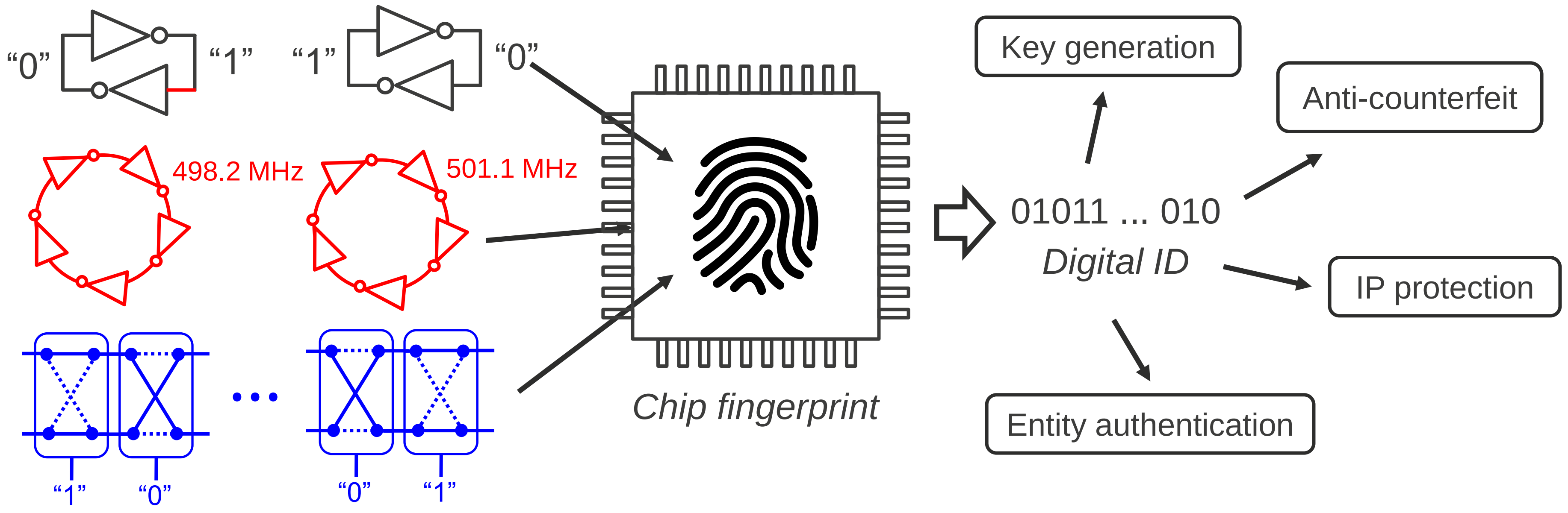
Sujoy Sinha Roy
sujoy.sinharoy@iaik.tugraz.at

Outline

- Introduction to PUFs
- Basic implementations
- Important PUF properties
- Design example
- Summary

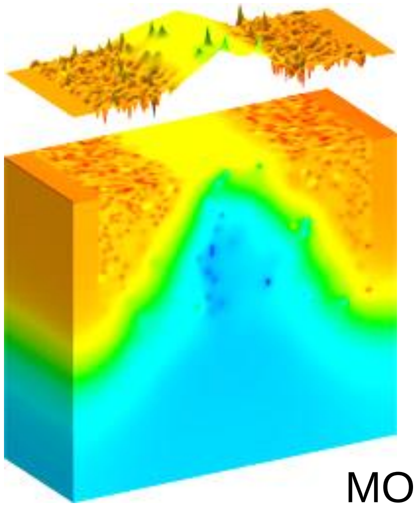
Silicon PUF: An unique fingerprint of a chip

- PUF can be viewed as a *unique* fingerprint of a chip
- Comes from *random process variations*
- Various implementations and applications

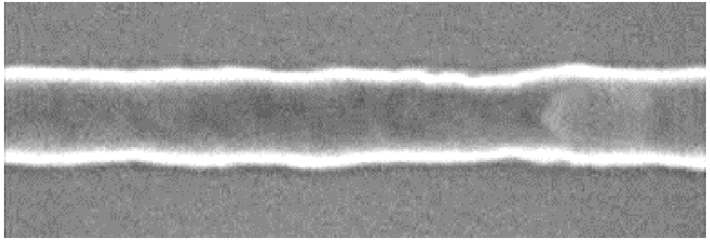


Variability is inherently presented in ICs

- Variability in transistors and interconnect
- In general undesired – except for PUFs
- Random dopant fluctuation
- Interconnect width is not always the same



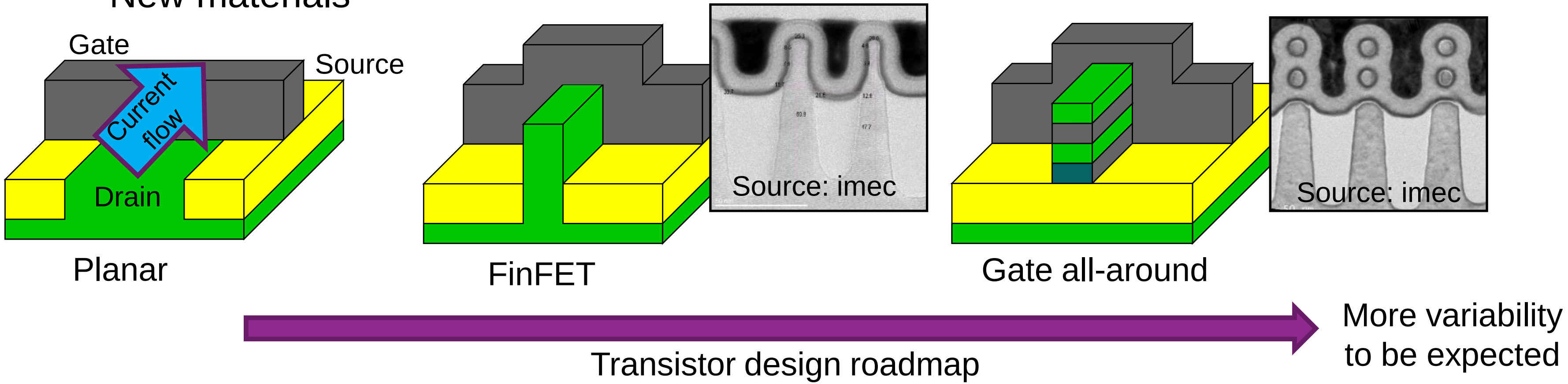
MOSFET



Interconnect

More opportunities brought by scaling

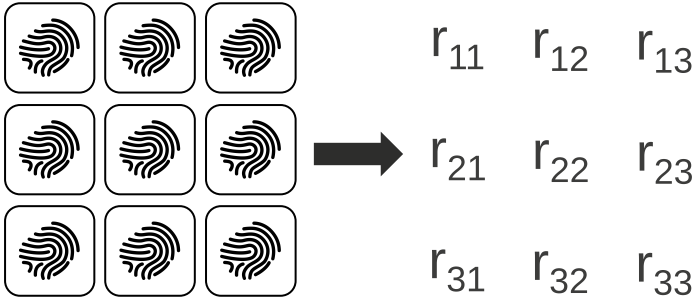
- Even more challenging to manufacture identical devices in scaled technologies
  - Moore’s Law
  - 40nm → 28nm →16nm → 7nm → ...
- More variability comes from:
  - More processing steps
  - Decreased size (e.g. 2nm difference → **5%** in 40nm and **30%** in 7nm)
  - New materials

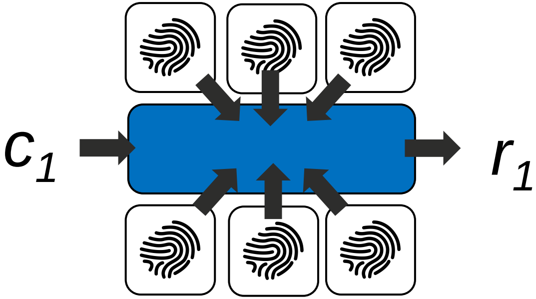


Outline

- Introduction to PUFs
- Basic implementations
- Important PUF properties
- Design example
- Summary

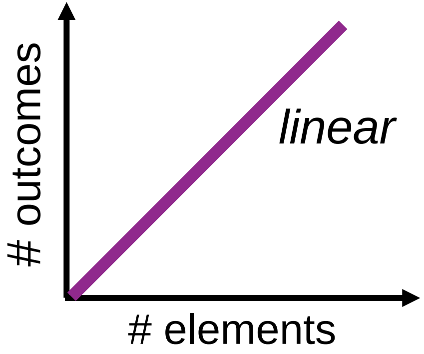
Two design methodologies

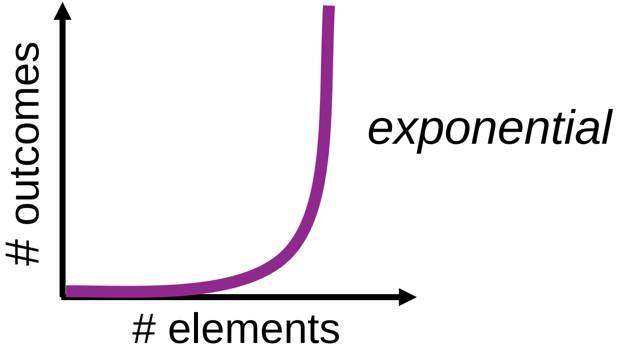




Weak PUF

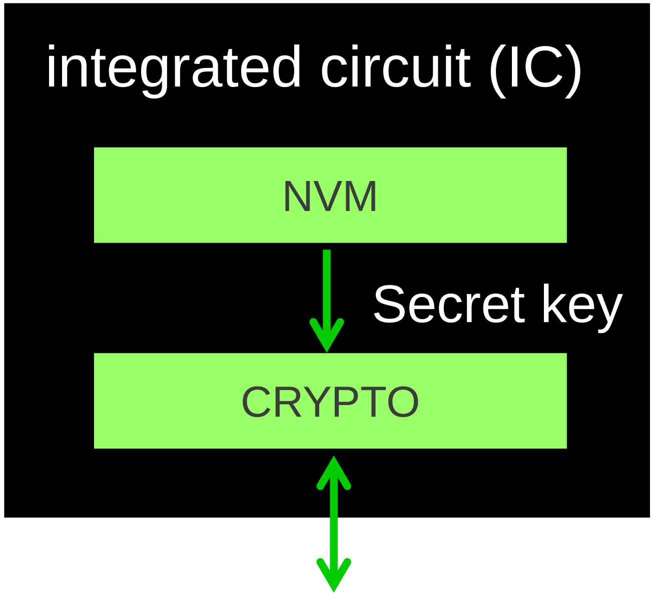
Strong PUF





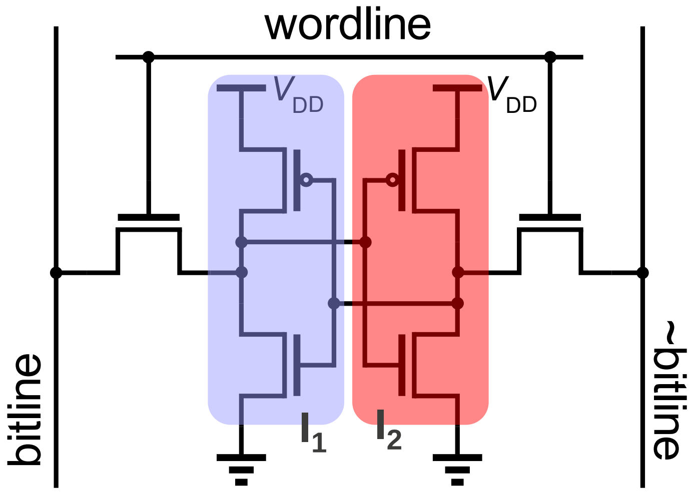
Replacing secure non-volatile memory

- The root key is typically stored in *secure* NVMs:
  - EEPROM/Flash
  - Fuses/Anti-fuses
  - Battery-backed SRAM
- Concerns:
  - Physical attacks
  - Resource constraints (cost)
- **PUF – generates its own unique key**

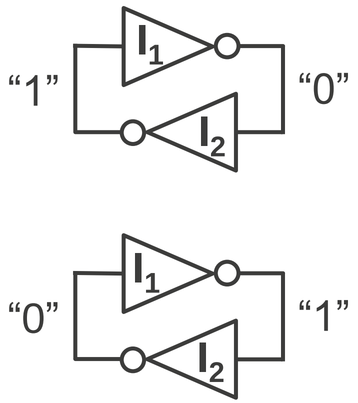


SRAM PUF – a classic weak PUF

- 2D array of 1-bit memory cells
- Variability: ***mismatch*** between the cross-coupled inverters
- Volatile: data is cleared after power-off



6T-SRAM cell

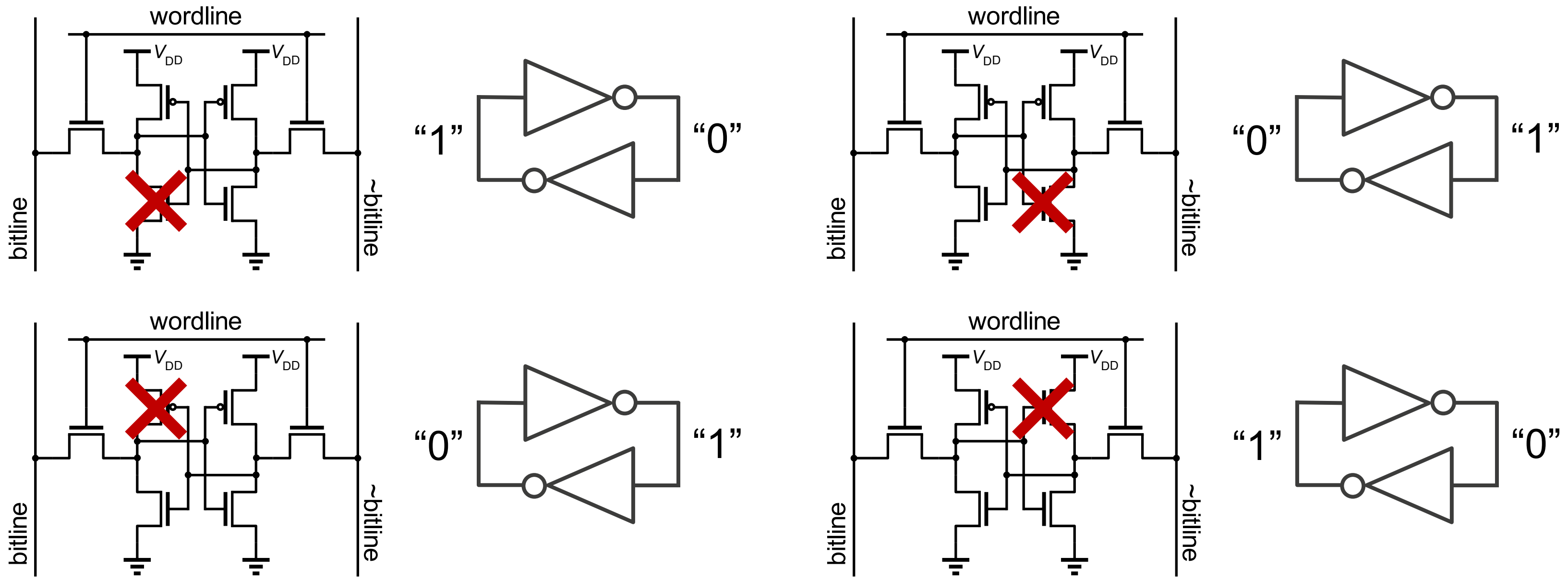


Bi-stable states

Two possible outcomes after power-up

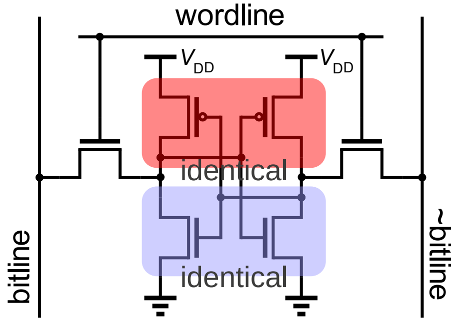
Transistor variations determines PUF bits

- Assume one of the transistors is much weaker than others
- Four extreme cases



Variations do not always lead to desired results

- If the variation is insignificant for a particular cell

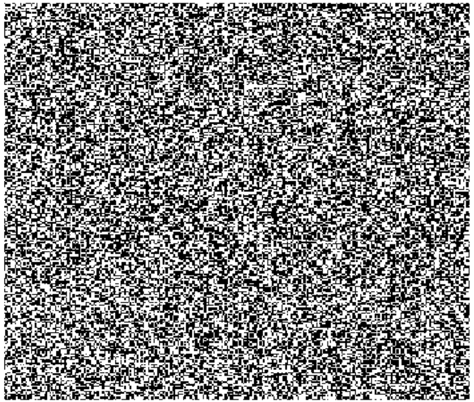


No preferred state

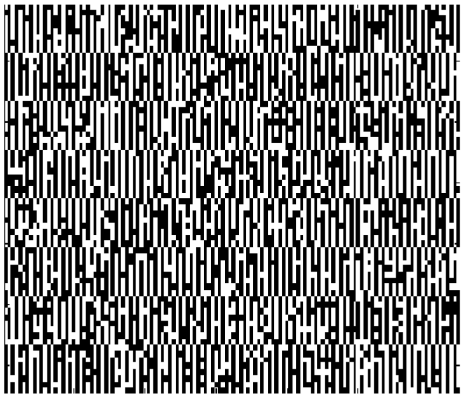
PUF bits determined by noise (RNG-like)

- If the variation is not completely random

white = 0
black = 1



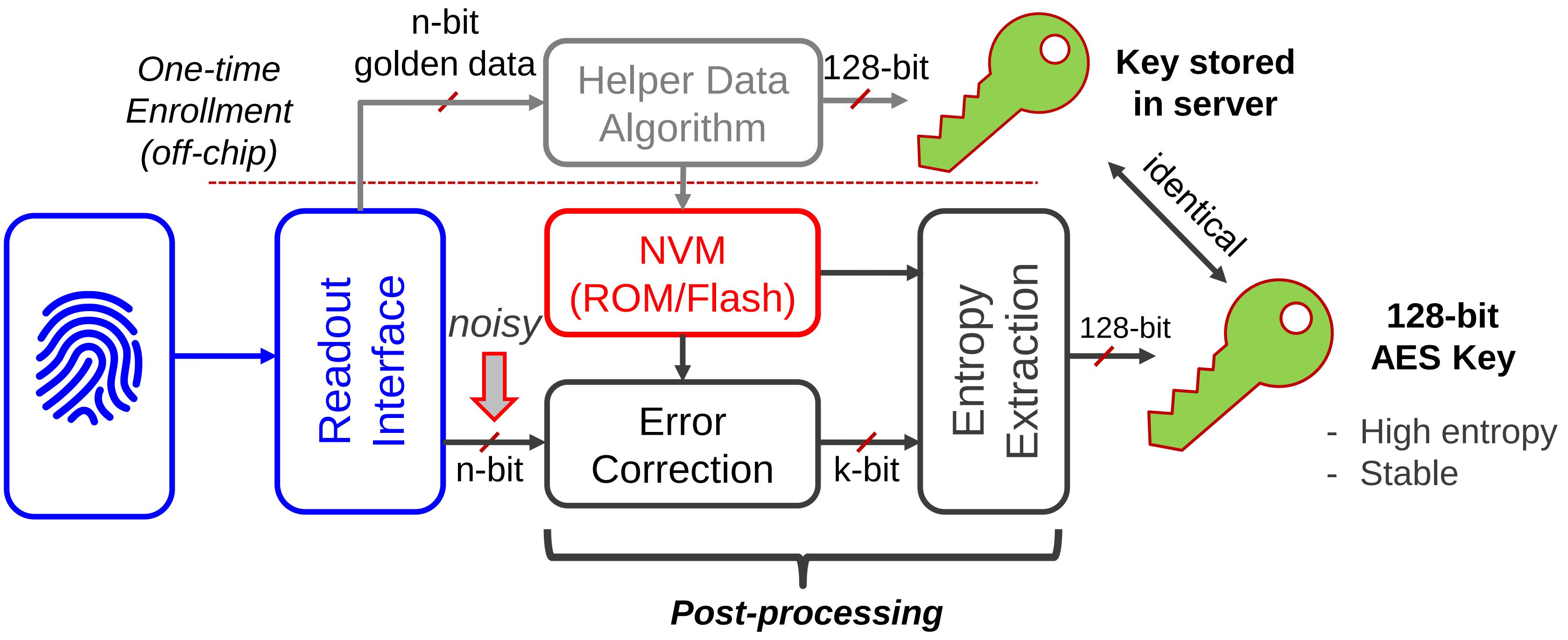
STM32-F100R8



PIC16F1825

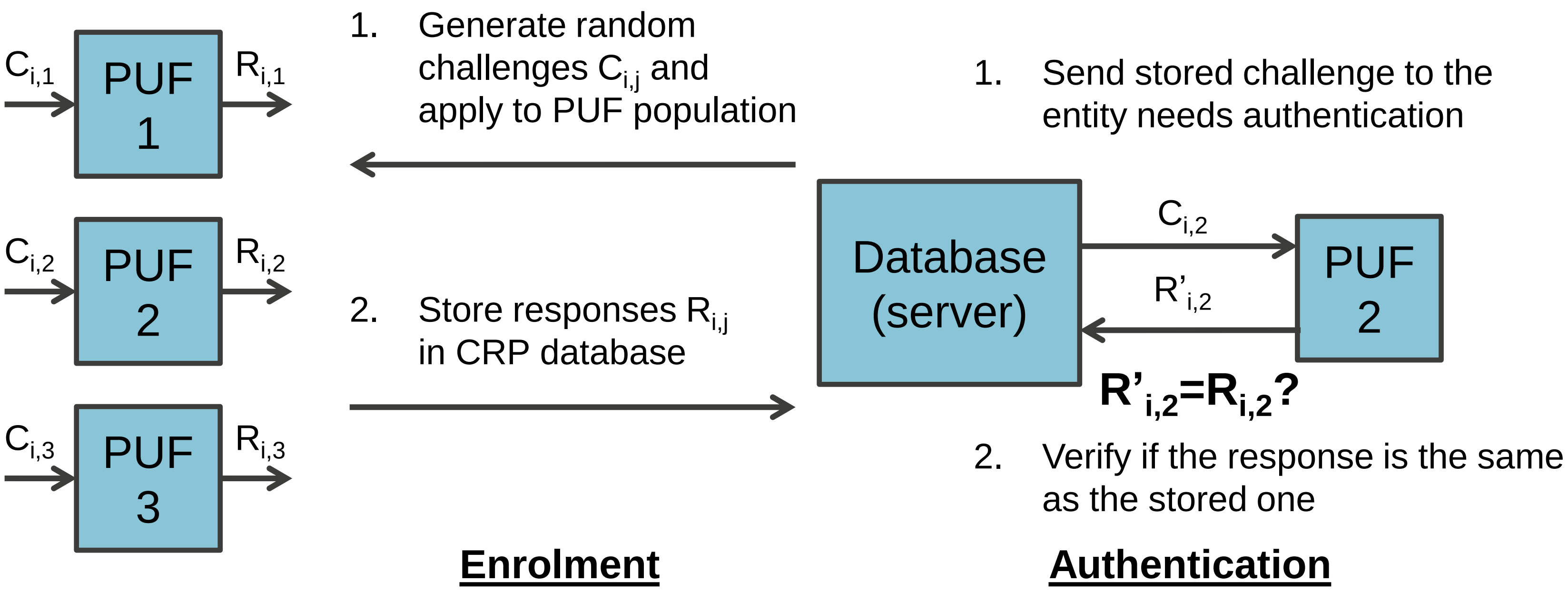
[Van Herrewege, TrustED 2013]

From process variation to a secret key



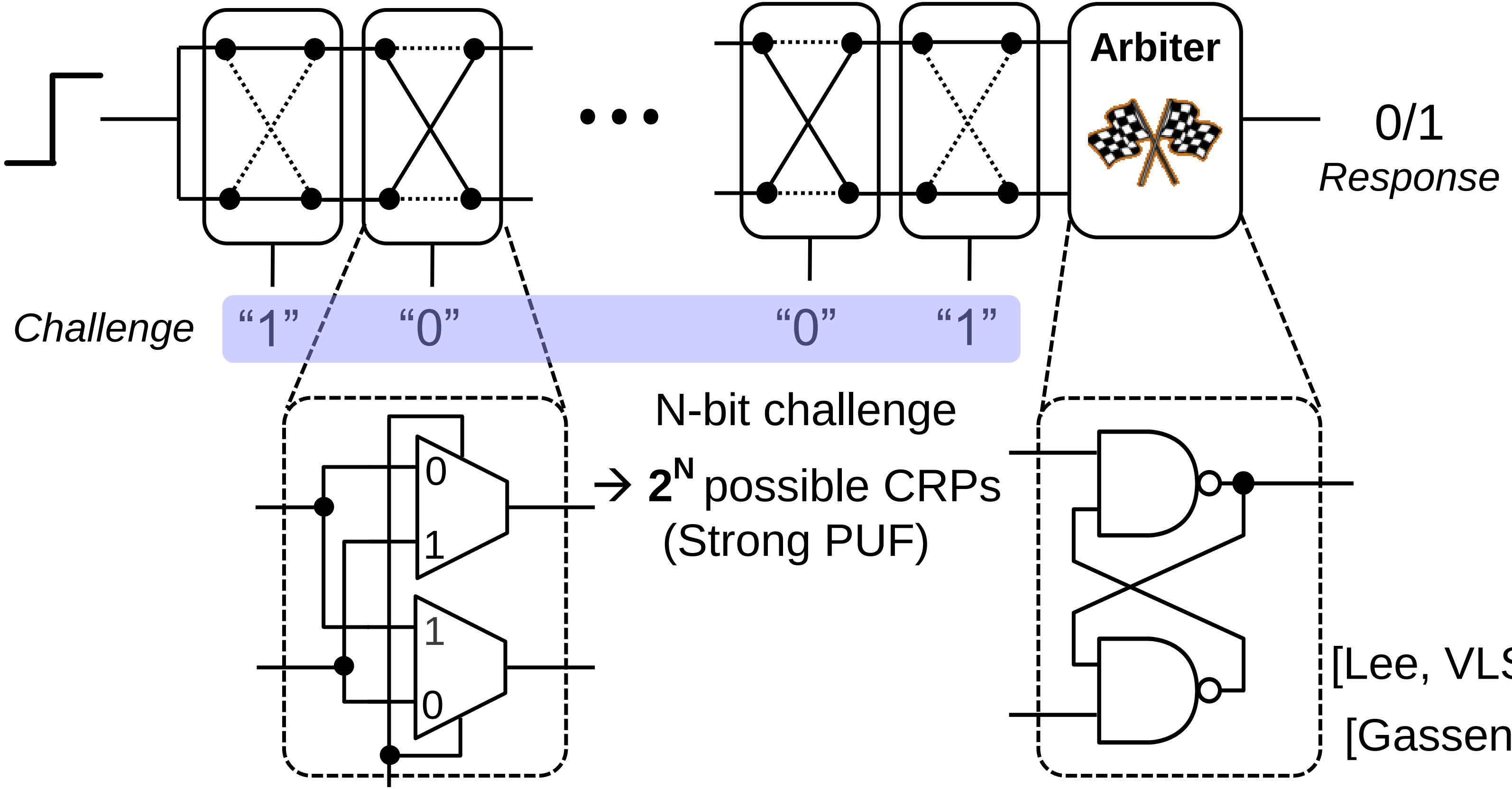
Realizing an ideal authentication scheme

- Entity authentication based on challenge and response



*Needs a huge amount of **uncorrelated** challenge-response pairs (CRPs)*

Arbiter PUF – based on timing differences



N-bit challenge

→ 2^N possible CRPs

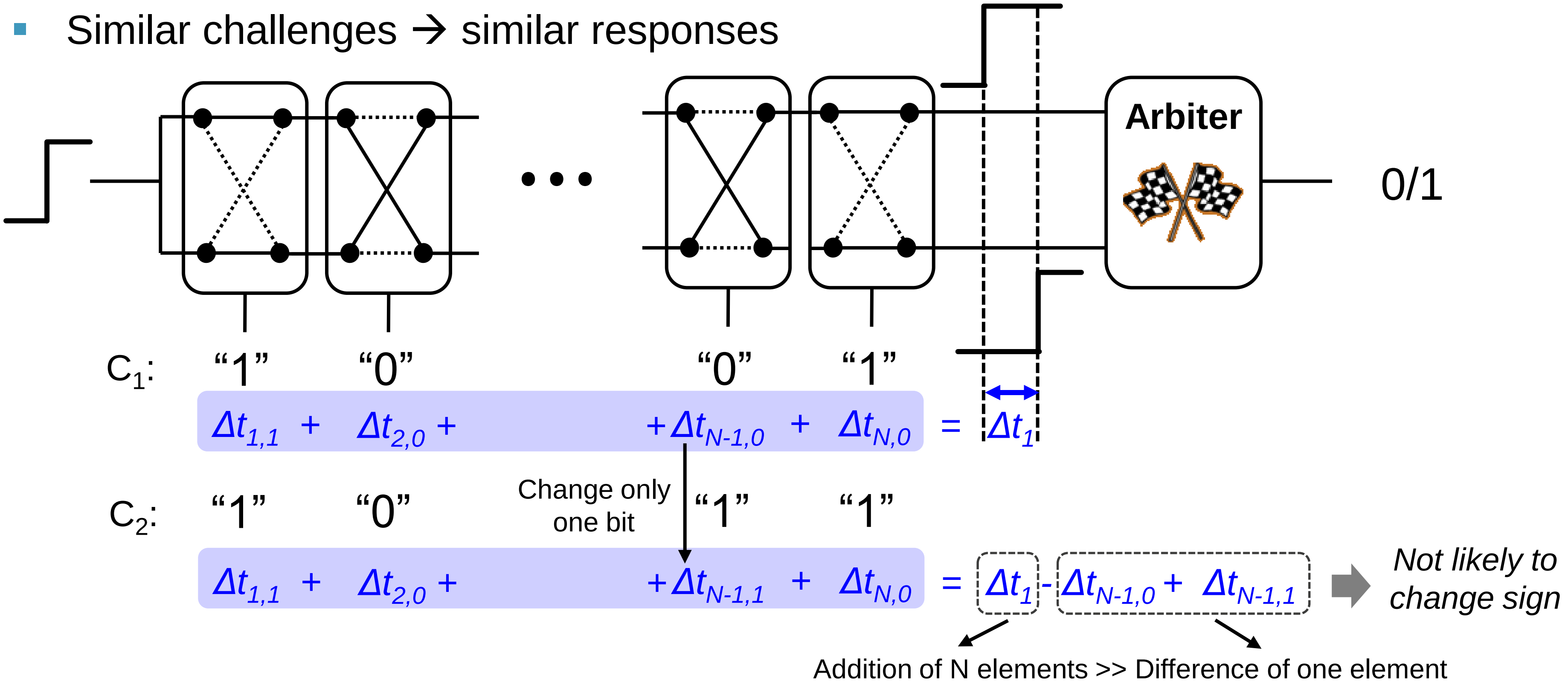
(Strong PUF)

[Lee, VLSIC 2004]

[Gassend, 2004]

Arbiter PUF is not an ideal strong PUF

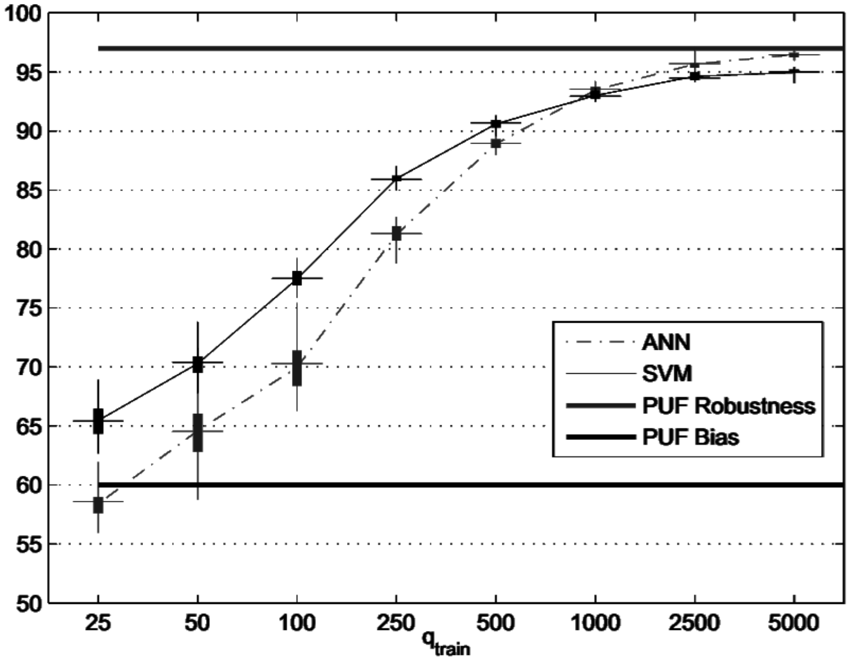
- *Linear* additive structure: sum of delays
- Similar challenges → similar responses



Responses can be easily predicted

- CRPs are highly correlated: low entropy

→ Prone to machine learning (ML) attacks



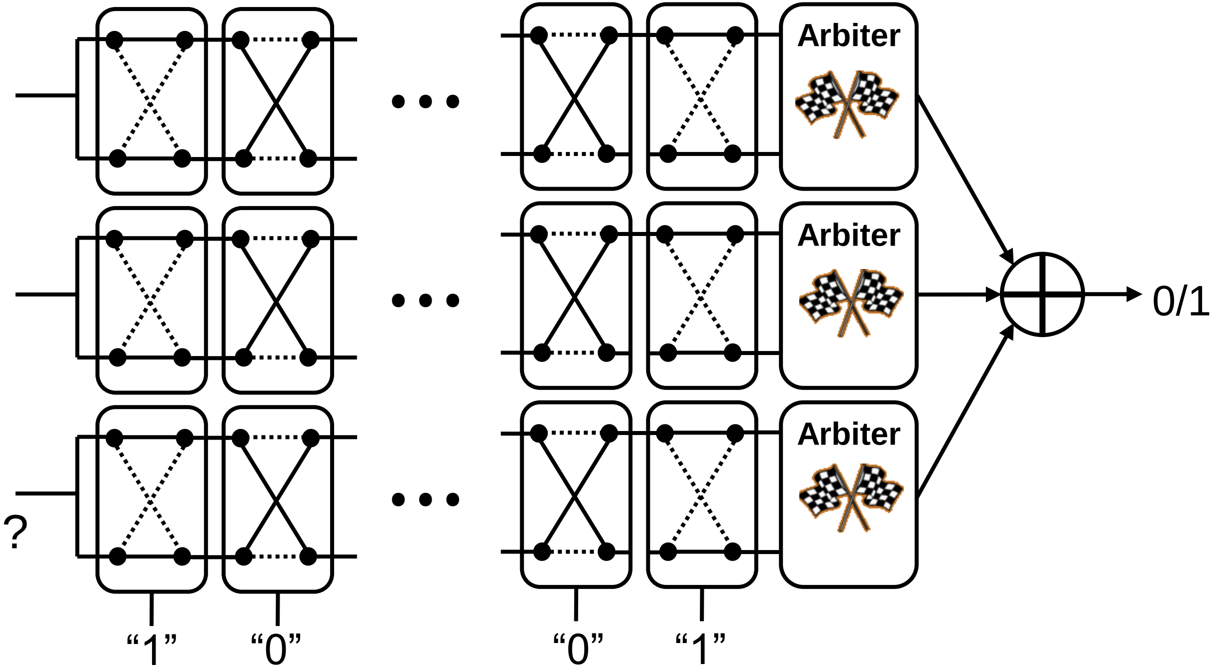
Experimental results on 65 nm CMOS: only a few 1000 CRPs are sufficient to model the PUF with high accuracy

[Hospodar, WIFS 2012]

[Ruhrmair, ACM CCS 2010]

Make it less predictable by XORing

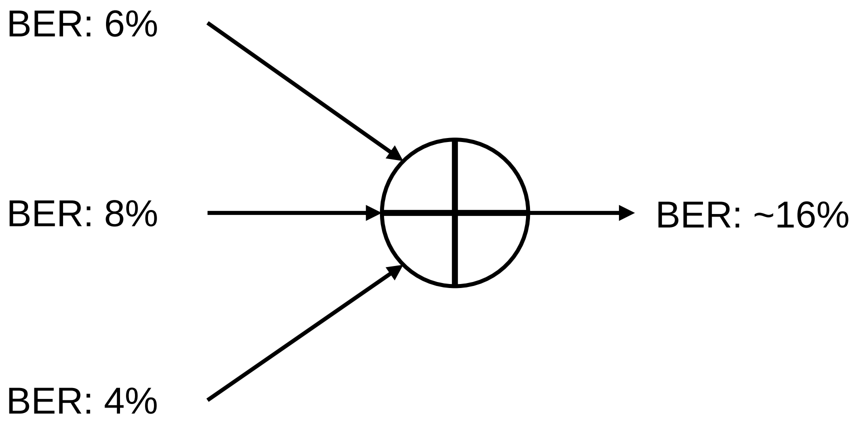
- XOR: non-linear operation
  - CRPs less correlated
  - → More CRPs for training
- More resilient to machine learning attacks
- Can we infinitely increase the number of XORs to make ML attacks infeasible?



Assume flip 1 challenge bit → 5% probability to flip response bit
XOR by 3 → **~14%**

of XORs is limited by noise

- Non-linear operation → **Noise amplification**



- Too many XORs → Too much noise
- Ends up behaving like **RNGs**

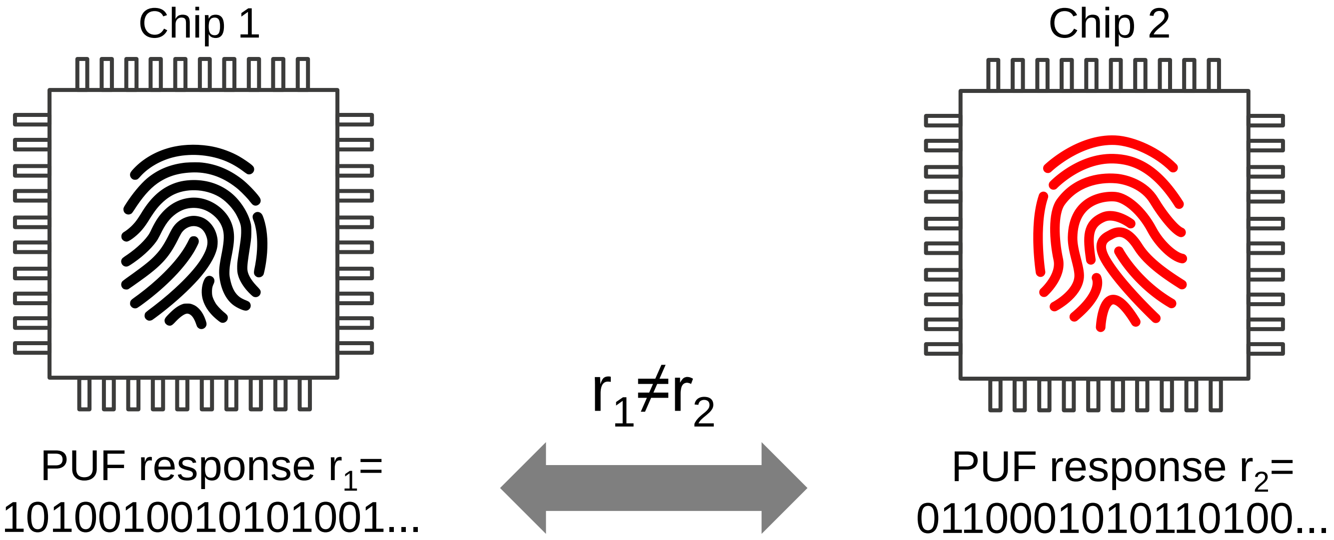
Is it possible to make an ideal strong PUF?

Outline

- Introduction to PUFs
- Basic implementations
- Important PUF properties
  - Uniqueness
  - Reliability (stability)
- Design example
- Summary

Uniqueness

- Two identically manufactured chips have different “fingerprint”
- Each chip has its *unique* PUF response



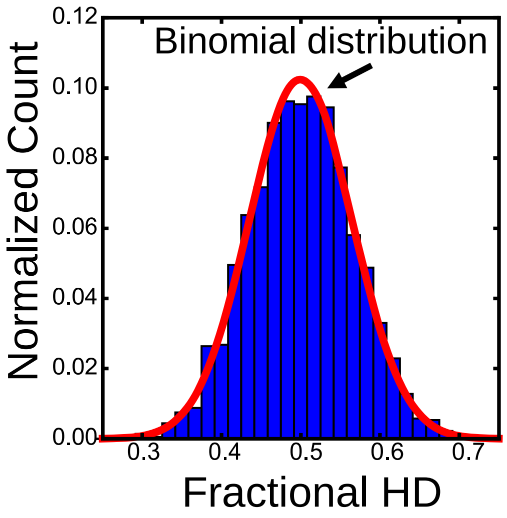
Estimate uniqueness by inter-distance

- Hamming distance, HD(r1, r2)
- Fractional-HD = HD(r1, r2) / n (n = # bits)
- Ideal-case: binomial distribution with success probability 0.5
  - Mean = **n/2** (50%)
  - Variance = n/4

r_1= 101011001010101001…
r_2= 011101010101110100…

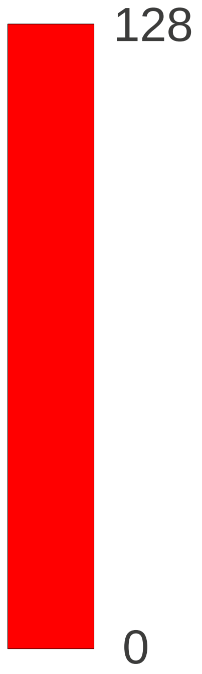
1 1 1 1 1 1 1 1

Sum=HD(r1, r2)

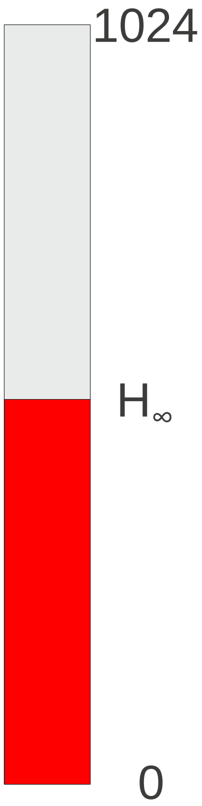


Min-entropy of a secret key

- E.g. 128-bit AES
- Key length = 128 bits
- Min-entropy = 128 bit
- Uniform distribution
- An attacker guesses the key first time right with probability: $\mathbf{2^{-128}}$



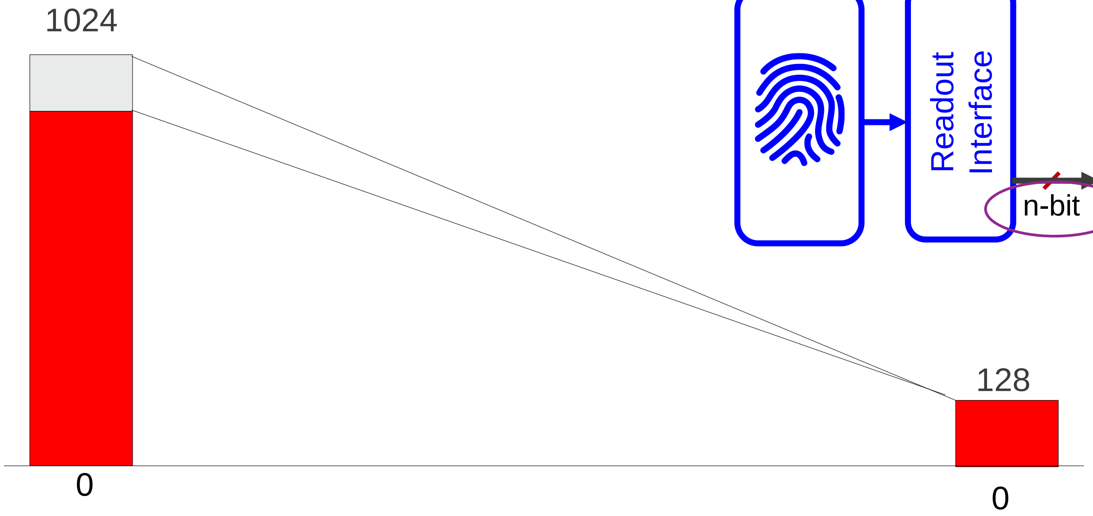
Min-Entropy of a PUF



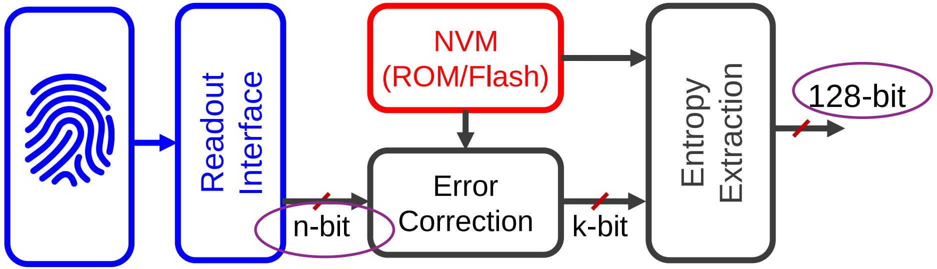
- Nearly impossible to determine exhaustively
  - Min-entropy tests require about 1M bits
  - Practically not feasible in a PUF, e.g., a 1024-bit SRAM PUF

- Can only get reasonably good estimation

From PUF to Secret Key



PUF-based key generator



PUF:
- Non-uniform
- Noisy

Helper Data Algorithm → Entropy loss

Key:
- Uniform
- Stable

Outline

- Introduction to PUFs
- Basic implementations
- Important PUF properties
  - Uniqueness
  - Reliability (stability)
- Design example
- Summary

Reliability

- PUF responses are not exactly reproducible
  - At different time
  - In different environment

PUF response r_1=

#1: 10100100101010001...

#2: 10110100001010001...

#3: 10100110101010001...

Short-term reliability (data stability)

- PUF response changed temporarily caused by:
  - Environment change (external)
  - Internal fluctuation

<u>External:</u>
- Temperature
- Supply voltage
- Humidity
- Radiation
- ...

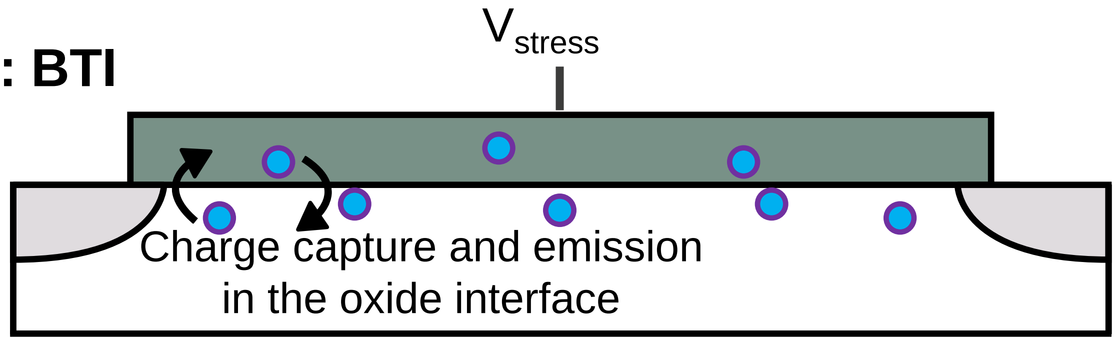
<u>Internal</u>
- White noise
- Flicker noise
- Cross-talk
- Glitch
- ...

How to improve the short-term reliability?

Long-term reliability

- Nearly permanent change caused by aging
  - Biased Temperature Instability (NBTI/PBTI)
  - Hot-carrier Injection (HCI)
  - Time-dependent dielectric breakdown (TDDB)
- Can be exploited to enhance the short-term reliability

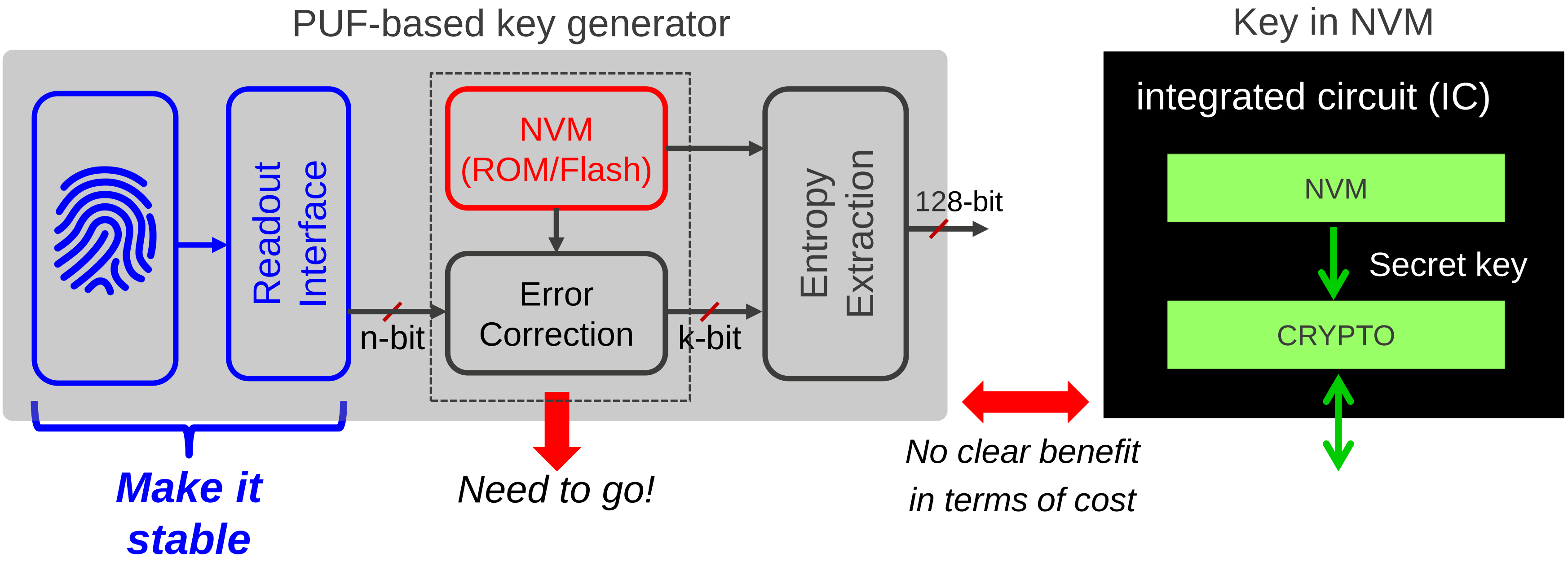
Example: BTI



→ V_T shift caused by charge trapping

Good reliability is crucial

- Error correction codes need to be stored → NVM needed
- Why not just store the key in NVM?



Outline

- Introduction to PUFs
- Basic implementations
- Important PUF properties
- Design example
  - Methods to improve data stability
- Summary

Methods to make PUF bits stable

- Error correction
  - Standardized mathematic operations → **Robust**
  - NVM is required
- Alternatives
  - Temporary majority voting
  - Dark-bit masking
  - Burn-in enhancement

Can achieve same robustness?

Reducing the effect of noise by averaging

- **Temporary majority voting (TMV)**:
  - Measure response bits multiple (N) times and output the most occurring value
- Reducing the error rate

#1: 1010010010101...

#2: 1011011000101...

#3: 1010011011101...

- - - - - - - - - - - - - - - - - -

TMV_3: 1010011010101...

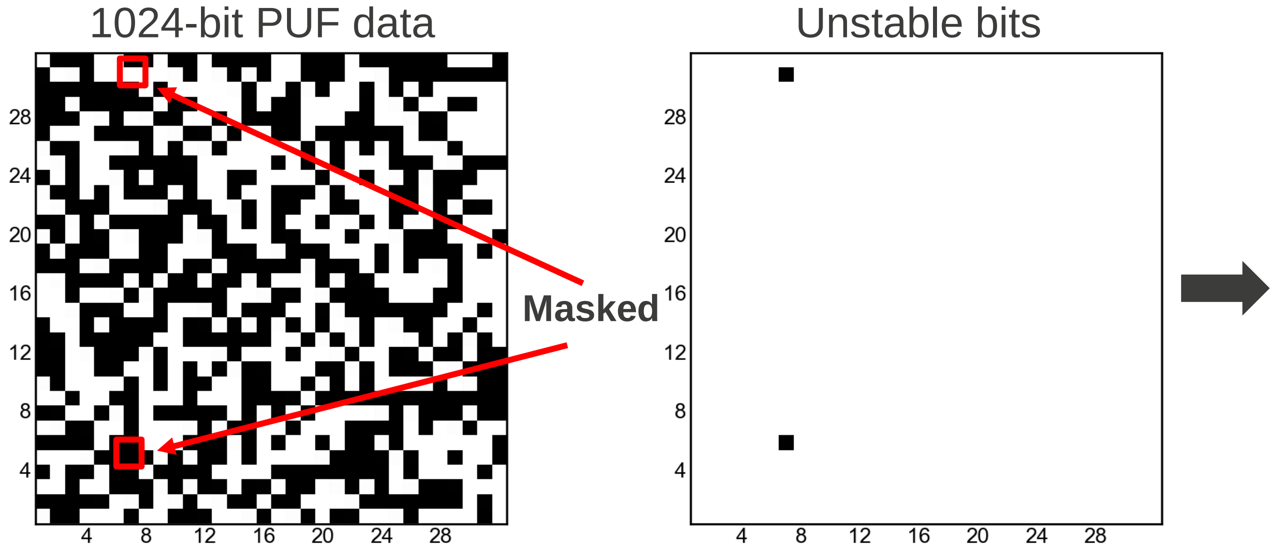
Error rate	1%	5%	10%	20%	30%	40%	45%	49%
N=3	3e-4	7.3e-3	2.8%	10.4%	21.6%	35.2%	42.5%	48.5%
N=5	1e-5	1.2e-3	8.6e-3	5.8%	16.3%	31.7%	40.7%	48.1%
N=11	<1e-9	5.8e-6	3.0e-4	1.2%	7.8%	24.7%	36.7%	47.3%
N=101	0	0	0	<1e-11	1.3e-5	2.1%	15.6%	42.0%

Not efficient for very noisy bits

- Need large N to ensure low error rate
- Large N → Large latency and needs more storage elements

Discarding all the noisy bits

- **Dark-bit masking**
  - Identify noisy bits and marked as “do not use”

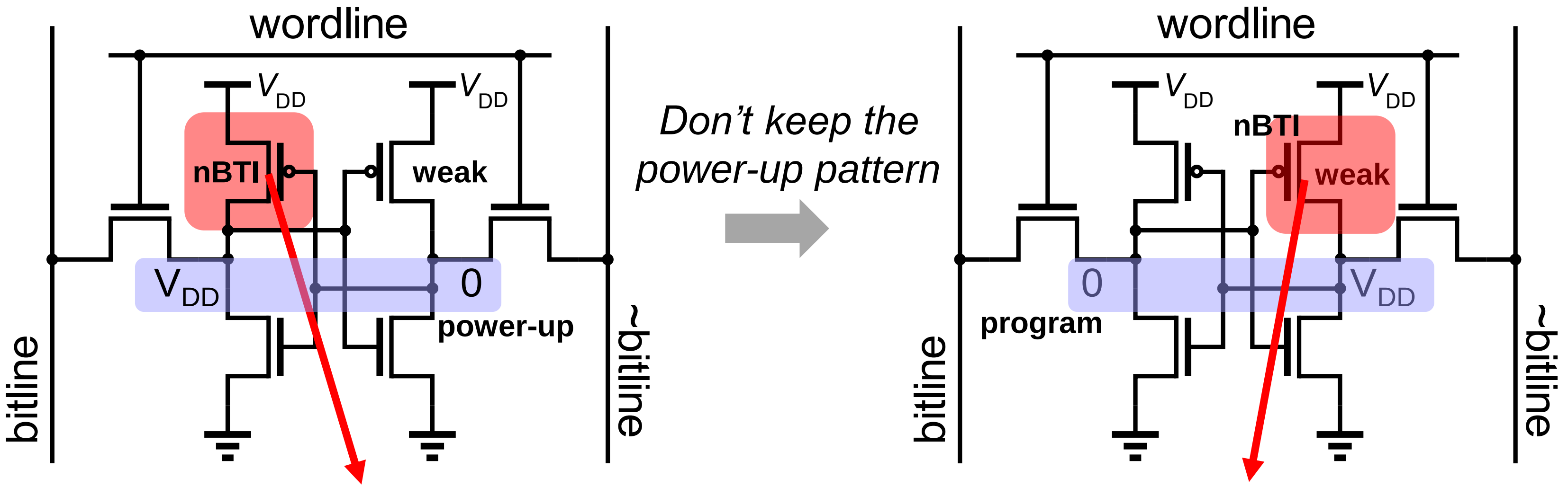


1022-bit
stable PUF data

- Two main concerns
  - How to identify unstable bits?
  - Still needs NVM to store mask information?

Exploit time dependent variability

- **Burn-in enhancement**
  - Apply intentional stress to age specific devices



Becomes weaker with time
→ ***Less difference***

Make it even weaker
→ ***More difference***

- **BTI**: Bias temperature instability is a degradation phenomenon affecting MOS
- **Concerns**: long stress time & recovery of degradation

Summary

- Silicon PUFs are unique fingerprints for chips
  - Benefits from process variation in silicon technology
- Secret key generation using weak PUFs
  - SRAM PUF as a classic example
  - Helper data algorithm is usually needed
- Entity authentication using strong PUFs
  - Arbiter PUFs can be used but is not ideal
  - Correlated CRPs are prone to ML attacks
- Uniqueness and reliability are the two key properties